STAT 4410/8416 Homework 4

Islam MD Tahidul

Due on Nov 12, 2023

1. Exploring XML data; In this problem we will read the xml data. For this we will obtain a xml data called olive oils from the link http://www.ggobi.org/book/data/olive.xml (http://www.ggobi.org/book/data/olive.xml). Please follow the directions in each step and provide your codes and output.

a. Parse the xml data from the above link and store in a object called `olive` . Answer the following questions using R code and type your answer:

```
library(XML)
url_ggobi <- "http://www.ggobi.org/book/data/olive.xml"
olive <-xmlParse(url_ggobi)
```

i. What is the name of the root of the xml file?

```
root_name <- XML::xmlName(xmlRoot(olive))
paste("The name of the root of the XML file is", root_name)
```

```
## [1] "The name of the root of the XML file is ggobidata"
```

ii. What is the count of data that is available under the root name?

```
count_data <- xmlSize(xmlRoot(olive))
count_data
```

```
## [1] 1
```

iii. Extract the text value for the `description` of the XML

```
# Find the description element using xpathSApply
description_element <- xpathSApply(olive, "//description", xmlValue)
# Print the text value of the description element
cat("Description:", description_element)
```

```
## Description:
## This is XML created by GGobi
```

b. Examine the actual file by going to the link above and answer the following questions using R code and type your answer:

i. Identify the path of `real variables` in the xml tree

```
rvPath <- "//ggobidata/data/variables/realvariable"
rvPath
```

```
## [1] "//ggobidata/data/variables/realvariable"
```

ii. What is the `names` of real variables?

```
    doc <- xmlParse(url_ggobi)
    real_vars <- getNodeSet(doc,rvPath)
    real_paths <- sapply(real_vars, function(x) {
      path <- character(0)
      while (!is.null(x)) {
        path <- c(xmlName(x), path)
        x <- xmlParent(x)
      }
      return(path)
    })

    # Extract the names and nicknames
    real_names <- sapply(real_vars, function(x) xmlGetAttr(x, "name"))
    cat("Real Variable Names: ", paste(real_names, collapse = ", "), "\n")
```

```
## Real Variable Names:  palmitic, palmitoleic, stearic, oleic, linoleic, linolenic,
arachidic, eicosenoic
```

iii. What is the count of the real variables?

```
    rvCount <-length(real_names)
    rvCount
```

```
## [1] 8
```

iv. Identify the path of `categorical variables` in the xml tree

```
    cvPath <- "//ggobidata/data/variables/categoricalvariable"
    cvPath
```

```
## [1] "//ggobidata/data/variables/categoricalvariable"
```

v. What is the `names` of categorical variables?

```
    cat_vars <- getNodeSet(doc,cvPath)
    # Extract the names of categorical variables
    cat_var_names <- sapply(cat_vars, function(x) xmlGetAttr(x, "name"))
    # Print the names of categorical variables
    cat("Names of Categorical Variables:", paste(cat_var_names, collapse = ", "),
"\n")
```

```
## Names of Categorical Variables: region, area
```

vi. What is the count of the categorical variables?

```
cvCount <- length(cat_var_names)
cvCount
```

```
## [1] 2
```

```
vii. How many levels does `categoricalvariable` with `name=area` have? Extract the te
xt value for level 5.
```

```
    # Use getNodeSet to extract the specified categoricalvariable
    area_cat_var <- getNodeSet(doc, "//variables/categoricalvariable[@name='area']")

    # Extract the count of levels for the specified categoricalvariable
    num_levels <- as.numeric(xmlGetAttr(area_cat_var[[1]], "count"))

    # Extract the text value for level 5
    level_5_text <- xpathSApply(doc, "//variables/categoricalvariable[@name='area']/l
evels/level[@value='5']", xmlValue)
    level_5_text
```

```
## [1] "Inland-Sardinia"
```

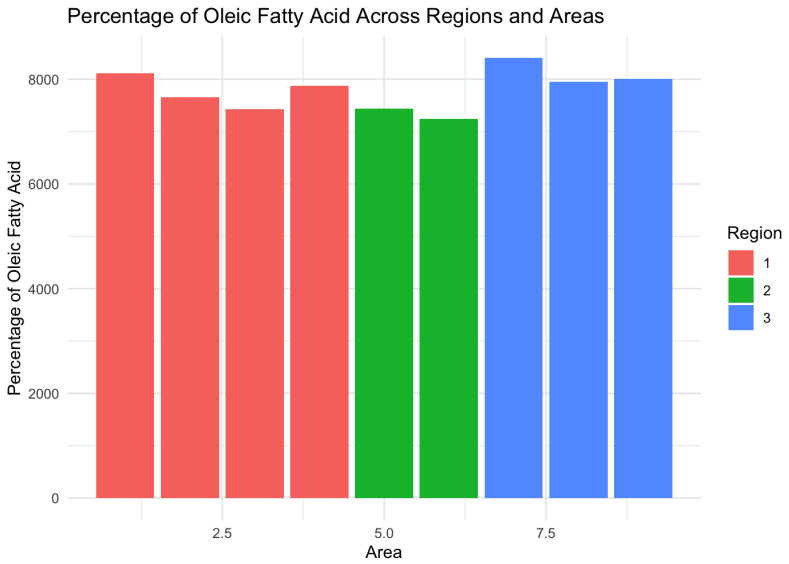
c. Notice the path for the data in xml file. Use that path to obtain the data and store the data in a data frame called `oliveDat` . Change the column names as you have obtained the column names. Display some data.

```
path_olive <- "//ggobidata/data/records/record"
# Find the XML nodes containing the data
record_nodes <- getNodeSet(olive, path_olive)
# Initialize empty lists for data
data_values <- list()
# Loop through record nodes to extract data
for (node in record_nodes) {
  # Extract the text values and split them into numeric values
  values <- as.numeric(unlist(strsplit(xmlValue(node), " ")))

  # Append data to the list
  data_values <- append(data_values, list(values))
}

# Combine data into a data frame
oliveDat <- as.data.frame(do.call(rbind, data_values))

col_names <- c("region", "area", "palmitic", "palmitoleic", "stearic", "oleic", "lino
leic", "linolenic", "arachidic", "eicosenoic")

# Assign the column names to the data frame
colnames(oliveDat) <- col_names
# Display the first few rows of the data frame
head(oliveDat,5)
```

```
##   region area palmitic palmitoleic stearic oleic linoleic linolenic arachidic
## 1      1    1     1075          75     226  7823      672        NA        NA
## 2      1    1     1088          73     224  7709      781        31        61
## 3      1    1      911          54     246  8113      549        31        63
## 4      1    1      966          57     240  7952      619        50        78
## 5      1    1     1051          67     259  7771      672        50        80
##   eicosenoic NA
## 1         60 29
## 2         29  1
## 3         29  1
## 4         35  1
## 5         46  1
```

d. Generate a plot of your choice to display any feature of `oliveDat` data. Notice that the column names are different fatty acids. The values are % of fatty acids found in the Italian olive oils coming from different regions and areas.

```
# Load required libraries
library(ggplot2)
library(dplyr)

# Extract relevant columns
plot_data <- oliveDat %>%
  select(region, area, oleic)

# Plotting
ggplot(plot_data, aes(x = area, y = oleic, fill = factor(region))) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Percentage of Oleic Fatty Acid Across Regions and Areas",
       x = "Area",
       y = "Percentage of Oleic Fatty Acid") +
  scale_fill_discrete(name = "Region") +
  theme_minimal()
```



2. Working with date-time data; The object `myDate` contains the date and time. Based on this object answer the following questions using R code and type the answer after your findings.

```
myDate <- "2022-10-01 09:42:43"
```

a. Convert `myDate` into a date and time object with Portland, OR time zone. Display the result.

```
library(lubridate)
# Convert myDate to POSIXct object with Omaha, NE time zone
myDate <- ymd_hms(myDate, tz = "PST")
myDate
```

```
## [1] "2022-10-01 17:42:43 GMT"
```

b. Write your codes so that it displays the week day and also the month of `myDate` .

```
weekday <- weekdays(myDate)
weekday
```

```
## [1] "Saturday"
```

```
month <- months(myDate)
month
```

```
## [1] "October"
```

c. What weekday and the month is it after exactly 100 years from `myDate` ?

```
futureDate <- myDate + years(100)
weekday_future <- weekdays(futureDate)
month_future <- months(futureDate)
weekday_future
```

```
## [1] "Thursday"
```

```
month_future
```

```
## [1] "October"
```

d. Add two month with myDate and display the resulting date time. Explain why the time zone has changed even though you did not ask for time zone change.

```
newDate <- myDate + months(2)
# Convert newDate to a character string for display
newDate_str <- format(newDate, "%Y-%m-%d %H:%M:%S")
newDate_str
```

```
## [1] "2022-12-01 17:42:43"
```

The change in time zone might occur due to the daylight saving time (DST) adjustment

e. Suppose this homework is due on November 21, 2023 by 11.59PM. Compute and display how many seconds you got to complete this homework? Also compute the hours.

```
# Load required library
library(lubridate)
# Current date and time
currentDateTime <- ymd_hms("2023-11-12 23:59:00", tz = "UTC")
# Deadline for the homework
deadline <- ymd_hms("2023-11-21 23:59:00", tz = "UTC")
# Calculate the time difference
timeRemaining <- deadline - currentDateTime
# Convert time difference to seconds
secondsRemaining <- as.numeric(timeRemaining, units = "secs")
# Convert seconds to hours
hoursRemaining <- round(secondsRemaining / 3600,0)
cat("\nSeconds:", secondsRemaining, "seconds")
```

```
##
## Seconds: 777600 seconds
```

```
cat("\nHours:", hoursRemaining, "hours")
```

```
##
## Hours: 216 hours
```

f. Suppose you are working with a Time-Series data. Where should the Time Value be? X-Axis or the Y-Axis? Explain your answer.

answer: In a time-series plot, the time variable is typically placed on the x-axis (horizontal axis), while the variable of interest (the variable being measured over time) is placed on the y-axis (vertical axis). The rationale behind this convention is to represent the chronological order of observations.

g. How do you get the current date as set in the computer? Does the date belong to a Leap Year?

```
current_date <- Sys.Date() # by this i can get current date
# Check if the current year is a leap year
is_leap_year <- leap_year(current_date)
is_leap_year
```

```
## [1] FALSE
```

h. For the years `2021 & 2022` , count the number of weekends. Which year has the highest number of Weekends?

```
# Define the start and end dates for the years 2021 and 2022
start_date_2021 <- as.Date("2021-01-01")
end_date_2021 <- as.Date("2021-12-31")

start_date_2022 <- as.Date("2022-01-01")
end_date_2022 <- as.Date("2022-12-31")

# Generate sequences of dates for each year
dates_2021 <- seq(start_date_2021, end_date_2021, by = "days")
dates_2022 <- seq(start_date_2022, end_date_2022, by = "days")

# Count the number of weekends for each year
weekends_2021 <- sum(weekdays(dates_2021) %in% c("Saturday", "Sunday"))
weekends_2022 <- sum(weekdays(dates_2022) %in% c("Saturday", "Sunday"))

# Print the results
cat("Number of weekends in 2021:", weekends_2021, "\n")
```

```
## Number of weekends in 2021: 104
```

```
cat("Number of weekends in 2022:", weekends_2022, "\n")
```

```
## Number of weekends in 2022: 105
```

```
cat("\n")
```

```
# Determine which year has the highest number of weekends
if (weekends_2021 > weekends_2022) {
  cat("2021 has the highest number of weekends.\n")
} else if (weekends_2021 < weekends_2022) {
  cat("2022 has the highest number of weekends.\n")
} else {
  cat("Both years have the same number of weekends.\n")
}
```

```
## 2022 has the highest number of weekends.
```

i. What is the month(MM) and day(DD) on the 305 th day of the current year?

```
# Get the current year
current_year <- year(Sys.Date())

# Calculate the date for the 305th day of the current year
target_date <- ymd(paste(current_year, "0101", sep = "")) + days(304)

# Extract the month and day
month_value <- month(target_date)
day_value <- day(target_date)

# Print the results
cat("Month (MM):", month_value, "\n")
```

```
## Month (MM): 11
```

```
cat("Day (DD):", day_value, "\n")
```

```
## Day (DD): 1
```

j. The Date `2022-10-20` is formatted `YYYY-MM-DD`, format it into `MM-DD-YYYY`

```
original_date <- as.Date("2022-10-20")
formatted_date <- format(original_date, "%m-%d-%Y")
cat("Formatted Date:", formatted_date, "\n")
```

```
## Formatted Date: 10-20-2022
```

k. Find if the Date on the question above is the weekend or a weekday.

```
day_of_week <- weekdays(original_date)

if (day_of_week %in% c("Saturday", "Sunday")) {
  cat("The date is a weekend.\n")
} else {
  cat("The date is a weekday.\n")
}
```

```
## The date is a weekday.
```

3. Creating HTML Page; In this problem we would like to create a basic HTML page. Please follow each of the steps below and finally submit your HTML file on Canvas. Please note that you don't need to answer these questions here in the .Rmd file.

a. Open a notepad or any plain text editor. Write down some basic HTML codes as shown in online (year 2014) Lecture 15, slide 6 and modify according to the following questions. Save the file as hw4.html and upload on Canvas as a separate file.
b. Write "What is data science?" in the first header tag, `<h1></h1>`
c. Hw1 solution contains the answer of what is data science. The answer has three paragraphs. Write the three paragraphs of text about data science in three different paragraph tags `<p></p>`. You can copy the text from hw1 solution.
d. Write "What we learnt from hw1" in second heading under tag `<h2></h2>`
e. Copy all the points we learnt in hw1 solution. List all the points under ordered list tag `<ol></ol>`. Notice that each item of the list should be inside list item tag `<li></li>`.
f. Now we want to make the text beautiful. For this we would write some CSS codes in between `<head></head>` tag under `<style></style>`. For this please refer to online (year 2014) lecture 15 slide 8. First change the fonts of the body tag to Helvetica Neue.
g. For the paragraph that contains the definition of data science, give an attribute `id='dfn'` and in CSS change the color of 'dfn' to white, background-color to olive and font to be bold.
h. For other paragraphs, give an attribute `class='cls'` and in CSS change the color of 'cls' to green.
i. Write CSS so that color of h1, h2 headers becomes `orange`.
j. (Optional and will not be graded) Write java Scripts codes so that onClick on `h1` header, it shows a message 'Its about data science'.

4. Walmart Sales Analysis Download and read the dataset `walmart_sales.csv` and `walmart_fuel_prices.csv` .

```
walmart_sale <- read.csv("walmart_sales.csv")
walmart_fuel_price <- read.csv("walmart_fuel_prices.csv")
```

We will follow the following data description when working with the above 2 datasets:

- `index:` index is a default value of count
- `Store:` Store is represented in number ID(1,2,3,4,…)
- `Dept:` Dept is Department in each Store represented in number ID (1,2,3,4,…)
- `Date:` Date is in YYYY-MM-DD char format - *needs to be converted into Date data type*
- `Weekly_Sales:` Sales of a given Dept in a given Store for the Date
- `Temperature:` Average temperature on the Date at given Store region
- `Fuel_Price:` Cost of the Fuel on the given Date at a given Store
- `IsHoliday:` Is the given Date a holiday Week?

Answer all of the following questions below and support your answer showing the codes and a plot (if applicable):

a. For both datasets, breakdown the `Date` column and create additional new columns `Year` , `Month` , and `Day` . You should now have additional 3 new columns in your both dataset. Report only the column names for both the dataset.

```
walmart_sale <- walmart_sale %>%
  mutate(Year = year(Date),
         Month = month(Date),
         Day = day(Date))

# View the modified dataset
colnames(walmart_sale)
```

```
## [1] "Store"        "Dept"         "Date"         "Weekly_Sales" "IsHoliday"
## [6] "Year"         "Month"        "Day"
```

```
walmart_fuel_price <- walmart_fuel_price %>%
  mutate(Year = year(Date),
         Month = month(Date),
         Day = day(Date))

# View the modified dataset
colnames(walmart_fuel_price)
```

```
## [1] "index"       "Store"       "Date"        "Temperature" "Fuel_Price"
## [6] "IsHoliday"   "Year"        "Month"       "Day"
```

b. In `walmart_sales` : which `Month(s)` of `Year` have the highest `Weekly_Sales` ? Report the Year, Month, Store, and Dept.

```
library(dplyr)
highest_sales <- walmart_sale %>%
  filter(Weekly_Sales == max(Weekly_Sales))
# Get the month name
month_name <- month.name[highest_sales$Month]
cat("Highest weekly sales in the year of", highest_sales$Year,
    ", in the month of", month_name,
    ", store is", highest_sales$Store,
    "and department is", highest_sales$Dept)
```

```
## Highest weekly sales in the year of 2010 , in the month of November , store is 10
and department is 72
```

c. In `walmart_sales` : calculate the average monthly sales by Department for each Store. Which Store(s) has the highest average monthly sales on the department(s)? Report the Store, Department, Date.

```
walmart_sale$Date  <- as.Date(walmart_sale$Date)

average_sales <- walmart_sale %>%
  group_by(Store, Dept, Month,Date) %>%
  summarise(AverageMonthlySales = mean(Weekly_Sales))

max_sales <- average_sales %>%
  group_by(Dept) %>%
  filter(AverageMonthlySales == max(AverageMonthlySales)) %>%
  arrange(-AverageMonthlySales)
paste("Highest average monthly sales date is ",max_sales$Date[1],
    ", store is", max_sales$Store[1],
    "and department is", max_sales$Dept[1])
```

```
## [1] "Highest average monthly sales date is  2010-11-26 , store is 10 and departmen
t is 72"
```

d. In `walmart_sales` : which month of year `2011` has the highest overall sales by Store? Name the holiday(Labor day, July 4th, Halloween, Thanksgiving, Christmas,… etcs) that falls on the month. After that do the same for `2012` . Does the highest sales per month fall on the same holiday for both years? Report your findings for both year.

```
year2011 <- walmart_sale %>%
  filter(Year == 2011)%>%
  group_by(Store,Month)%>%
  summarise(TotalSales = sum(Weekly_Sales))%>%
  arrange(-TotalSales)
#head(year2011,1)
year2012 <- walmart_sale %>%
  filter(Year == 2012)%>%
  group_by(Store,Month)%>%
  summarise(TotalSales = sum(Weekly_Sales))%>%
  arrange(-TotalSales)
#head(year2012,1)

paste("in year 2011,December is the highest selling month that means Christmas has an
impact, otherwise 2012 june is the highest selling month")
```

```
## [1] "in year 2011,December is the highest selling month that means Christmas has a
n impact, otherwise 2012 june is the highest selling month"
```

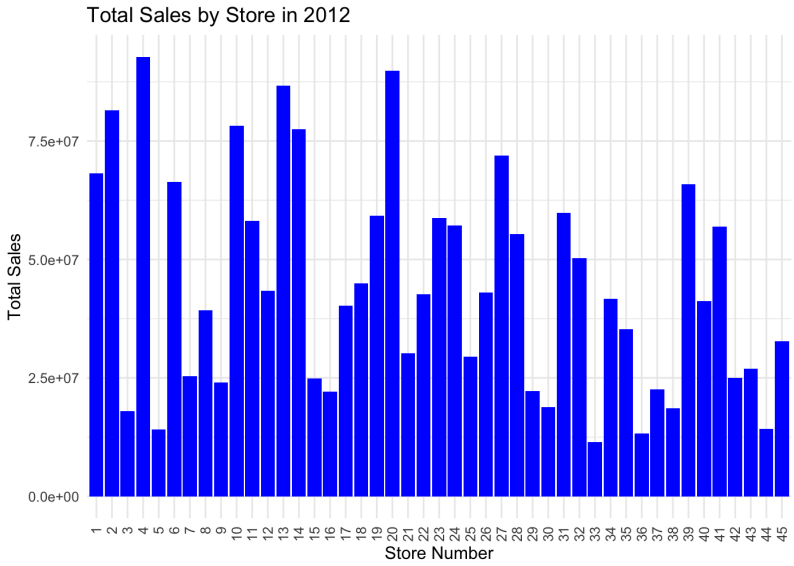
e. In `walmart_sales` : report the lowest sales per month for the year `2011` for `IsHoliday == TRUE` . Name the holiday(Labor day, July 4th, Halloween, Thanksgiving, Christmas,… etcs) that falls on the month. Do the same for `2012` and report if the lowest sales are on the same month.

```
year2011_lowest <- walmart_sale %>%
  filter(Year == 2011 & IsHoliday == TRUE )%>%
  group_by(Store,Month)%>%
  summarise(TotalSales = sum(Weekly_Sales))%>%
  arrange(TotalSales)

#head(year2011_lowest,1)

year2012_lowest <- walmart_sale %>%
  filter(Year == 2012 & IsHoliday == TRUE )%>%
  group_by(Store,Month)%>%
  summarise(TotalSales = sum(Weekly_Sales))%>%
  arrange(TotalSales)

#head(year2012_lowest,1)
paste("Only in 2011 ,December has lowest sales where only Christmas is holiday,on the
other hand 2012 has lowest sales on February")
```

```
## [1] "Only in 2011 ,December has lowest sales where only Christmas is holiday,on th
e other hand 2012 has lowest sales on February"
```

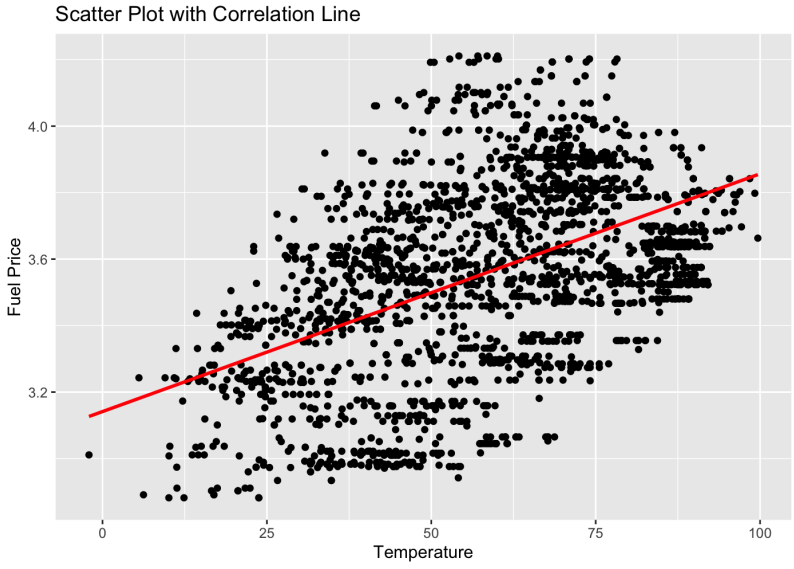
f. In `walmart_sales` : We have 45 unique stores. Generate a nice plot on the total sales by store for the year `2012` . Report the Store number.

```
library(ggplot2)
# Assuming walmart_sales is your data frame
walmart_sale$Date <- as.Date(walmart_sale$Date)
# Filter data for the year 2012
sales_2012 <- walmart_sale[walmart_sale$Year == 2012, ]
# Calculate total sales by store
total_sales_by_store <- sales_2012 %>%
  group_by(Store) %>%
  summarise(TotalSales = sum(Weekly_Sales))
# Create a bar plot with more space on the x-axis
ggplot(total_sales_by_store, aes(x = factor(Store), y = TotalSales)) +
  geom_bar(stat = "identity", fill = "blue", width = 0.9) +  # Adjust width as needed
  labs(title = "Total Sales by Store in 2012",
       x = "Store Number",
       y = "Total Sales") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1))
```



g. In `walmart_fuel_prices` : For the year `2011` do you think higher the `temperature` relates to higher `fuel price` ? Support your answer with a nice plot.

```
# Assuming your dataset is named walmart_fuel_prices
library(ggplot2)
# Filter data for the year 2011
walmart_fuel_2011 <- subset(walmart_fuel_price, Year == 2011)
correlation_coefficient <- cor(walmart_fuel_2011$Temperature, walmart_fuel_2011$Fuel_
Price)
ggplot(walmart_fuel_2011, aes(x = Temperature, y = Fuel_Price)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, col = "red") +
  labs(title = "Scatter Plot with Correlation Line",
       x = "Temperature",
       y = "Fuel Price")
```




```
cat("Correlation coefficient:", correlation_coefficient, "So we can say that there is
moderate effect on Temperature and fuel price")
```

```
## Correlation coefficient: 0.4882504 So we can say that there is moderate effect on
Temperature and fuel price
```

h. In `walmart_fuel_prices` : For the year `2010` which `Store` had the lowest Fuel Price? Report the `month` and `temperature` . On the same `month` , what was the highest fuel price for the store? Report the difference.

```
# Assuming your dataset is named walmart_fuel_prices
# Convert Date to Date format
walmart_fuel_price$Date <- as.Date(walmart_fuel_price$Date)
# Filter data for the year 2010
walmart_fuel_2010 <- subset(walmart_fuel_price, Year == 2010)
# Find the store with the lowest fuel price in each month
lowest_price_by_month <- walmart_fuel_2010 %>%
  group_by(Month) %>%
  slice(which.min(Fuel_Price))

# Find the highest fuel price for the store in the same month
highest_price_by_month <- walmart_fuel_2010 %>%
  inner_join(lowest_price_by_month, by = c("Month", "Store")) %>%
  group_by(Month) %>%
  slice(which.max(Fuel_Price.y))
#highest_price_by_month
# Calculate the difference in fuel prices
price_difference <- highest_price_by_month$Fuel_Price.x[1] - lowest_price_by_month$Fu
el_Price[1]
# Print the results
paste("Lowest price is in month number",lowest_price_by_month$Month[1],"and Temperatu
re is ", lowest_price_by_month$Temperature[1],"In this month highest fuel price is ",
highest_price_by_month$Fuel_Price.x[1],".By the way price difference is",price_differ
ence)
```

```
## [1] "Lowest price is in month number 2 and Temperature is  45.66 In this month hig
hest fuel price is  2.545 .By the way price difference is 0.073"
```

i. In `walmart_fuel_prices` : For the `IsHoliday == TRUE` , which month has the lowest `Fuel Price` for the year `2012` ? name the holiday(Labor day, July 4th, Halloween, Thanksgiving, Christmas,… etcs) that falls on the month. Also report month of the highest fuel price and name of the holiday.

```
# Assuming your dataset is named walmart_fuel_prices
# Convert Date to Date format
walmart_fuel_price$Date <- as.Date(walmart_fuel_price$Date)

# Filter data for the year 2012 and IsHoliday == TRUE
walmart_fuel_2012_holidays <- subset(walmart_fuel_price, Year == 2012 & IsHoliday ==
TRUE)

# Find the month with the lowest fuel price
lowest_price_month <- walmart_fuel_2012_holidays %>%
  filter(Fuel_Price == min(Fuel_Price)) %>%
  select(Month)

# Find the month with the highest fuel price
highest_price_month <- walmart_fuel_2012_holidays %>%
  filter(Fuel_Price == max(Fuel_Price)) %>%
  select(Month)

# Map month numbers to month names
month_name <- function(month_number) {
  months <- c("January", "February", "March", "April", "May", "June",
              "July", "August", "September", "October", "November", "December")
  return(months[month_number])
}

# Get holiday names for the lowest and highest fuel price months
lowest_price_holiday <- month_name(lowest_price_month$Month[1])
highest_price_holiday <- month_name(highest_price_month$Month[1])

# Print the results
cat("Month with the lowest fuel price for holidays in 2012:", lowest_price_holiday,
"\n")
```

```
## Month with the lowest fuel price for holidays in 2012: February
```

```
cat("Month with the highest fuel price for holidays in 2012:", highest_price_holiday)
```

```
## Month with the highest fuel price for holidays in 2012: September
```

5. Optional for undergraduate but mandatory for graduate students Download the data from Github - click here (https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_confirmed_US.csv)

```
git_link <- "https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_co
vid_19_data/csse_covid_19_time_series/time_series_covid19_confirmed_US.csv"
covid_data <- read.csv(git_link)
#write.csv(covid_data,"covid.csv",row.names = FALSE)
```

The link above contains a time-series data for COVID-19 confirmed cases in the US. Limit the data to only use `Nebraska State` and please answer the following questions:

a. What is the total confirmed cases in Nebraska as of October 30th 2020 as per the dataset?

```
nebraska_data <- covid_data %>%
  filter(Province_State == "Nebraska")
sum_X10.30.20 <- nebraska_data %>%
  select(contains("X10.30.20")) %>%
  summarize(total_cases = sum(X10.30.20))
sum_X10.30.20$total_cases
```

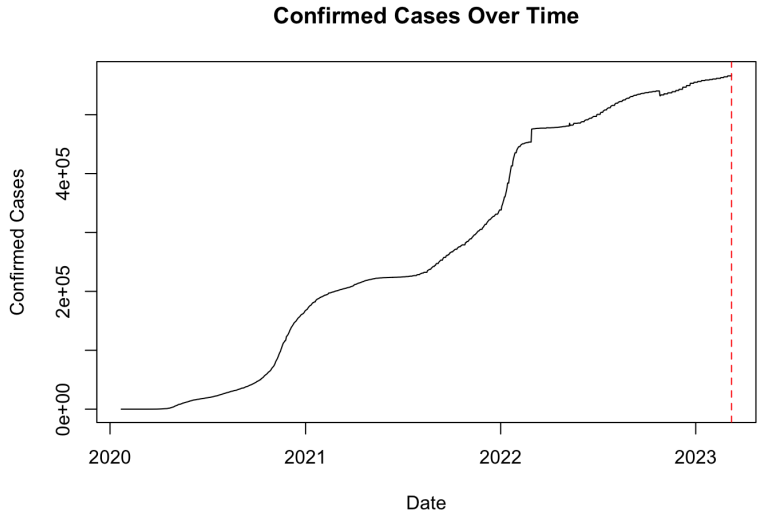
```
## [1] 69645
```

b. On what date has the highest confirmed cases? Demonstrate using a suitable graph for all the available data.

```
date_columns <- nebraska_data[, 12:ncol(nebraska_data)]

# Convert the column names to Date format
date_names <- as.Date(gsub("X", "", names(date_columns)), format = "%m.%d.%y")

# Sum the confirmed cases for each date
total_cases_per_date <- colSums(date_columns, na.rm = TRUE)

# Find the date with the highest confirmed cases
date_highest_cases <- date_names[which.max(total_cases_per_date)]
cat("The day when the highest confirmed cases is:", format(date_highest_cases, "%Y-%m
-%d"), "\n")
```

```
## The day when the highest confirmed cases is: 2023-03-09
```

```
# Plotting the data
plot(date_names, total_cases_per_date, type = "l", xlab = "Date", ylab = "Confirmed C
ases",
     main = "Confirmed Cases Over Time")
# Adding a vertical line to indicate the date with the highest cases
abline(v = as.numeric(date_highest_cases), col = "red", lty = 2)
```



c. Which County has the highest daily confirmed cases? Report both the County name and the date

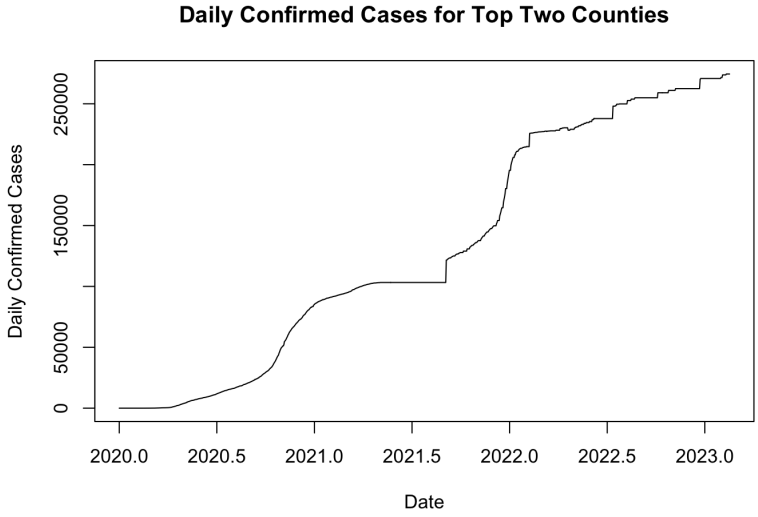
```
library(tidyr)
nebraska_data_long <- nebraska_data %>%select(-c(Lat, Long_, UID, iso2, iso3, code3,F
IPS,Country_Region,Province_State,Combined_Key)) %>%pivot_longer(cols = -c(Admin2), n
ames_to = "Date", values_to = "Cases")%>%
  mutate(Date = gsub("X", "", Date))
# Group by Admin2 to find the county with the highest daily confirmed cases
highest_cases_county <- nebraska_data_long %>%
  group_by(Admin2) %>%
  summarise(MaxCases = max(Cases, na.rm = TRUE)) %>%
  top_n(1, MaxCases)
highest_cases_answer<-highest_cases_county$Admin2
# Extract the county name with the highest daily confirmed cases and the associated d
ate
highest_county_name <- highest_cases_county$Admin2
date_highest_cases_county <- nebraska_data_long %>%
  filter(Admin2 == highest_county_name, Cases == max(nebraska_data_long$Cases)) %>%
  top_n(1,Cases)
# Display the county with the highest daily confirmed cases and the date
print(paste("County with the highest daily confirmed cases:", highest_cases_answer,"a
nd Date is ", date_highest_cases_county$Date[1]))
```

```
## [1] "County with the highest daily confirmed cases: Douglas and Date is  3.3.23"
```

d. Identify two counties that have top total confirmed cases. Generate a time series plot of daily confirm cases for these two counties.

```
highest_two_cases_county <- nebraska_data_long %>%
  group_by(Admin2) %>%
  summarise(MaxCases = max(Cases, na.rm = TRUE)) %>%
  top_n(2, MaxCases)


highest_two_cases_county <- nebraska_data %>%
  filter((Admin2 == "Douglas" | Admin2 == "Lancaster"))

# Extract data for the top two counties
top_two_counties_data <- nebraska_data %>%
  filter(Admin2 %in% highest_two_cases_county$Admin2)

# the date columns start from the 12th column
date_columns <- top_two_counties_data[, 12:ncol(top_two_counties_data)]

# Sum the confirmed cases for each date for the top two counties
daily_cases_top_two_counties <- colSums(date_columns, na.rm = TRUE)
# Convert the date column to Date format
dates <- gsub("X", "", names(daily_cases_top_two_counties))

# Convert to time series for plotting
daily_cases_ts <- ts(daily_cases_top_two_counties, start = c(2020, 1), frequency = 36
5.25)
```

```
# Create a time series for the daily confirmed cases
daily_cases_ts <- ts(daily_cases_top_two_counties, start = c(2020, 1), frequency = 36
5.25)

# Plot the time series
plot(daily_cases_ts, type = "l",
     xlab = "Date", ylab = "Daily Confirmed Cases",
     main = "Daily Confirmed Cases for Top Two Counties")
```




```
#!!!!!!!!!!!!!!!!!!!!!!
```

e. Show the total confirmed cases for all the locations in an interactive world map (hint: you may use `leaflet` package in `R`.

```
library("leaflet")
# Assuming your data frame is named covid_data
map_data <- nebraska_data
# Calculate total confirmed cases for each location
map_data$total_cases <- rowSums(map_data[, grepl("^X", names(map_data))], na.rm = F)

# Create a leaflet map
world_map <- leaflet(map_data) %>%
  addTiles() %>%
  addMarkers(
    ~Long_,
    ~Lat,
    popup = ~paste("Location: ", Combined_Key, "<br>Total Confirmed Cases: ", total_c
ases)

  )
# Display the map
world_map
```


Leaflet (https://leafletjs.com) | © OpenStreetMap
(https://openstreetmap.org/copyright/), ODbL
(https://opendatacommons.org/licenses/odbl/)